PROJEKTOVÁNÍ IS

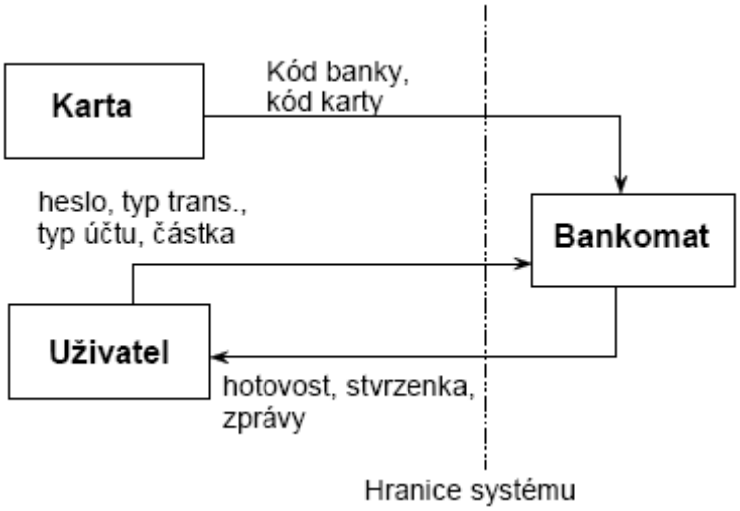
P10
2007-12-06

FUNKČNÍ MODELOVÁNÍ:

- ✓ Funkční model (FM) ukazuje, **jak** jsou počítány hodnoty, bez ohledu na posloupnost rozhodování nebo strukturu objektů.
- ✓ DFD je vhodná technika pro zobrazení funkčních závislostí jednotlivých dat.
- ✓ **Proces** v DFD odpovídá **aktivitám** nebo **akcím** v SD.
- ✓ **Toky** v DFD odpovídají **objektům** nebo hodnotám atributů v OD.
- ✓ Je vhodné vytvářet FM až **po** OM a DM.
- ✓ FM vytvoříme v následujících **krocích**:
  - Identifikujeme vstupní a výstupní hodnoty.
  - Vytvoříme DFD zobrazující funkční závislosti.
  - Popíšeme funkce.
  - Identifikujeme omezení.
  - Určíme kriteria optimálnosti.

Identifikace vstupních a výstupních hodnot:

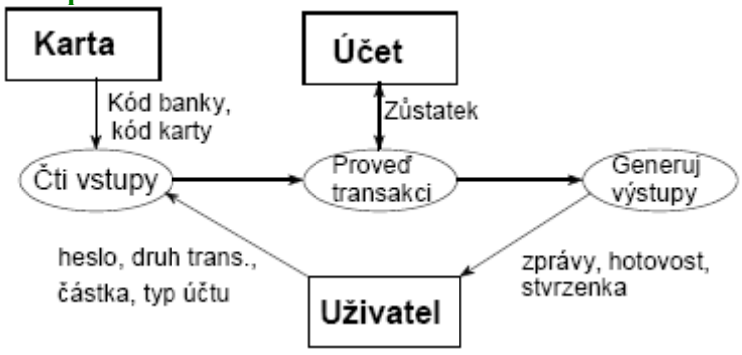
- ✓ Začneme od seznamu vstupních a výstupních hodnot (tj. parametrů událostí mezi systémem a okolím.



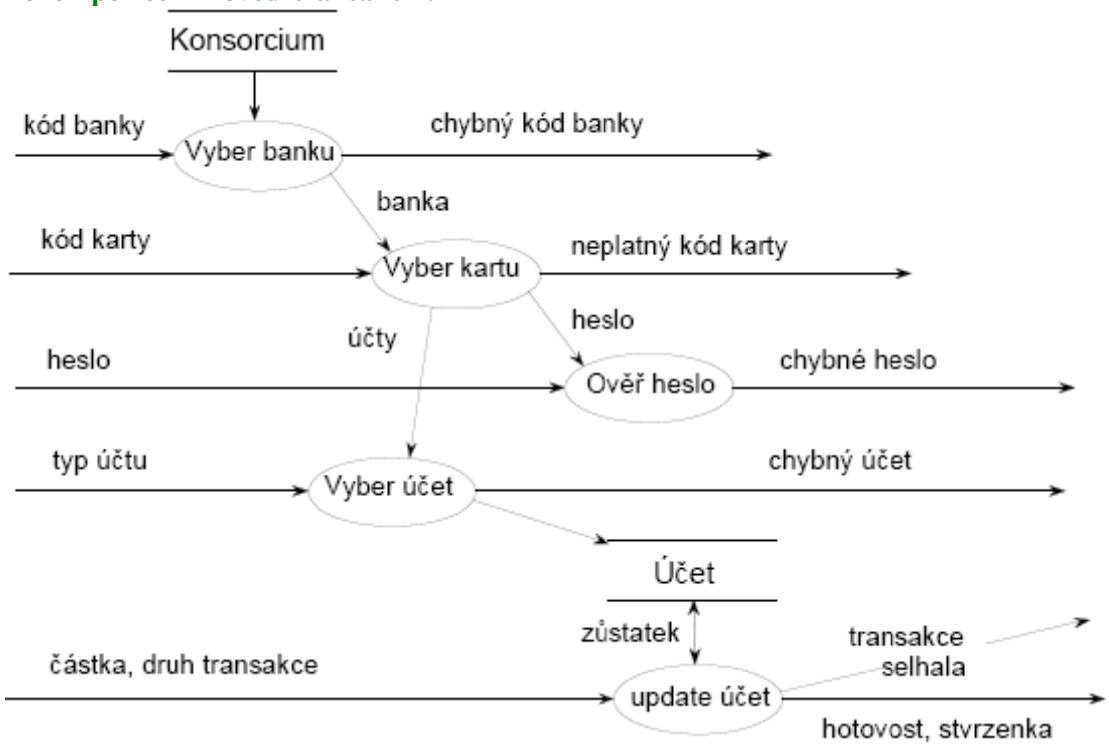
Vytvoření DFD:

- ✓ V DFD ukážeme, jak se získá (počítá) každá výstupní hodnota ze vstupních hodnot.
- ✓ Je možná **dekompozice** – vícevrstvá hierarchie.
- ✓ Je snazší postupovat **od výstupních hodnot k funkcím**, než od vstupních k výstupním hodnotám.

DFD pro Bankomat:



Dekompozice "Proveď transakci":



Popis funkcí:

- ✓ Po vyjasnění DFD proveďme popis každé funkce (přir. jazykem, mat. rovnicí, pseudokódem, ...)
- ✓ Zaměřme se na to, co funkce dělá, nikoliv jak ji implementovat.
- ✓ Popis může být deklarativní nebo procedurální.
  Např.: popis funkce "Update účet"
  - Update účet(účet, částka, druh transakce) -> hotovost, stvrzenka, zpráva
  - Když částka přesahuje zůstatek, odmítni transakci a nevydej hotovost.
  - Když částka nepřesahuje zůstatek, odečti částku od zůstatku a vydej hotovost.
  - Když...

Omezení mezi objekty:

- ✓ Omezení jsou **funkční** závislosti, které nejsou vstupně/výstupními závislostmi.
- ✓ **Prekondice** (předpoklady) na funkci jsou omezení, která musí splňovat vstupní hodnoty.
- ✓ **Postkondice** jsou požadavky na výstupní hodnoty.

Iterace analýzy:

- ✓ Analýzu musíme obvykle projít opakovaně, než je úplná (než úplně chápeme problém).
- ✓ Mezi analýzou a návrhem není žádná ostrá dělící čára.
- ✓ Neváhejme provést v modelu nezbytné změny, přesto, že jsme ho již "dokončili".
- ✓ Konečný model musí být ověřen objednatelem.
- ✓ Nutno si nechat potvrdit úpravy k požadavkům.

Souhrn k analýze:

- ✓ Nejdřív zapíšeme úvodní formulaci problému (může být neúplná, víceznačná nebo chybná – je to pouze začátek).
- ✓ **OM** ukazuje statickou strukturu skutečnosti. Postupně určíme tř. objektů, vazby, atributy, dědičnost. Informaci OM doplníme stručným textovým popisem.
- ✓ **DM** ukazuje chování systému. Přes scénáře vytvoříme SD každého aktivního objektu.
- ✓ **FM** ukazuje funkční odvození hodnot. Počínaje identif. vstup. a výstup. hodnot vytvoříme DFD.

Systémový návrh:

- ✓ Systémový návrh je vrcholová strategie pro rozložení na subsystémy a hlavní politická a koncepční rozhodnutí, ze kterých vyjde podrobný návrh.
- ✓ Celkové uspořádání systému do komponent (subsystémů) se nazývá architektura.
- ✓ Různé architektury kladou různý důraz na 3 modely.
- ✓ Uvedená pravidla platí také pro neobjektové postupy.
- ✓ Během návrhu se rozhoduje JAK bude problém řešen, postupně do větších detailů.
- ✓ Po vrcholovém rozhodnutí pro celý systém, může další práce pokračovat nezávisle pro jednotlivé subsystémy.
- ✓ Návrhář musí provést následující rozhodnutí:
  - Uspořádat systém do subsystémů
  - Rozpoznat konkurenčnost problému.
  - Přiřadit subsystémy procesorům a úlohám.
  - Vybrat způsob řízení úložišť dat (storů).
  - Zvládnout přístup ke globálním zdrojům.
  - Zvládnout hraniční podmínky.
- ✓ Často vybíráme architekturu na základě podobnosti s předchozími systémy.
- ✓ Jisté druhy architektury jsou vhodné pro řešení široké třídy problémů.

Rozklad systému na subsystémy:

- ✓ Prvním krokem v návrhu systému je rozklad na malý počet složek - komponent.
- ✓ Každá hlavní komponenta se nazývá subsystém.
- ✓ Rozklad se provádí podle společné vlastnosti: podobné funkčnosti, stejného fyzického umístění,...
- ✓ Subsystém není ani objekt ani funkce ale skupina tříd, vazeb, operací, událostí,..., které mají malý interface s ostatními subsystémy.
- ✓ Subsystém je obvykle určen službami, které poskytuje.
- ✓ Každý subsystém má dobře definovaný interface se zbytkem systému.
- ✓ Interface určuje tok informací přes hranice subsystému, ale ne vnitřní provedení.
- ✓ Každý subsystém může být proto navrhován nezávisle, aniž ovlivní ostatní.
- ✓ Subsystém musí být definován tak, aby většina interakcí nastala uvnitř subsystému, nikoliv přes jeho hranice.
- ✓ Počet subsystémů musí být malý. (20 je moc).
- ✓ Každý subsystém lze dále rozkládat na své vlastní subsystémy.
- ✓ Subsystém nejnižší úrovně (list hierarchie) je modul.
- ✓ Vztah mezi dvěma subsystémy může být klientdodavatel (client-suplier), nebo rovný s rovným peer-to-peer.
- ✓ Klient žádá dodavatele o službu, kterou tento provede a odevzdá výsledek; - jednostranná interakce.
- ✓ Klient musí znát interface dodavatele, ale dodavatel nezná interface svých klientů.
- ✓ Ve vztahu peer-to-peer každý subsystém může volat kterýkoliv jiný; - oboustranná interakce.
- ✓ Komunikace je zde obtížnější, protože si všichni musí znát navzájem své interface.
- ✓ Proto se snažíme o rozklad typu client-suplier – je jednodušší.
- ✓ Rozklad na subsystémy může probíhat horizontálně (vrstvy), nebo vertikálně (partitions)